J.W. PRICE 949/261.8433
Takakazu SHIOMI et al
NAKI-B082

S.N. 09/854,268
#3

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年　５月１５日

出願番号
Application Number: 特願２０００－１４１４９２

出願人
Applicant (s): 松下電器産業株式会社

２００１年　１月１９日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２０００－３１１２８９５

【書類名】　特許願
【整理番号】　2022520223
【提出日】　平成12年 5月15日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/46
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　塩見　隆一
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　葉山　悟
【発明者】
　【住所又は居所】　広島県東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内
　【氏名】　平本　建志
【特許出願人】
　【識別番号】　000005821
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100097445
　【弁理士】
　【氏名又は名称】　岩橋　文雄
【選任した代理人】
　【識別番号】　100103355
　【弁理士】
　【氏名又は名称】　坂口　智康

【選任した代理人】

【識別番号】　100109667

【弁理士】

【氏名又は名称】　内藤　浩樹

【手数料の表示】

【予納台帳番号】　011305

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9809938

【書類名】　明細書

【発明の名称】　アプリケーション実行装置

【特許請求の範囲】

【請求項１】　アプリケーションを記憶するアプリケーション記憶手段と、

前記アプリケーション記憶手段が記憶するアプリケーションを実行するアプリケーション実行手段と、

アプリケーションが使用する資源の回収の指示に使用される通知手段と、

前記通知手段を登録され、前記通知手段を用いて資源回収のタイミングを通知する資源回収通知手段と、

システムが備えるデバイスを制御し、アプリケーション実行手段が実行するアプリケーションに資源を提供し、前記通知手段を前記資源回収通知手段に登録し、前記資源回収通知手段より資源回収のタイミングを通知され、資源を回収するライブラリを備えることを特徴とするアプリケーション実行装置。

【請求項２】　問い合わせに応じて前記アプリケーション実行手段が実行するアプリケーションを特定するアプリケーション特定手段を備え、

前記資源回収通知手段は、資源回収通知時に資源回収の対象であるアプリケーションを通知し、

前記ライブラリは、アプリケーションに提供する資源を前記アプリケーション特定手段によって特定されたアプリケーションと対応付けて保持し、前記資源回収通知手段から資源回収の通知を受けた際、同時に通知される資源回収対象のアプリケーションに提供した資源だけを回収することを特徴とする請求項１記載のアプリケーション実行装置。

【請求項３】　前記資源回収通知手段は、資源回収通知時に資源回収の対象であるアプリケーションが複数あるとき、これら複数のアプリケーションを同時に通知することを特徴とする請求項２記載のアプリケーション実行装置。

【請求項４】　アプリケーション毎に、前記資源回収通知手段を備えるアプリケーション情報管理手段と、

問い合わせに応じて前記アプリケーション実行手段が実行するアプリケーションを特定するアプリケーション特定手段を備え、

前記アプリケーション情報手段が備える前記資源回収通知手段は、前記アプリケーション情報管理手段が管理するアプリケーションから資源回収を行う場合だけ、登録されている通知手段を用いて資源回収のタイミングを通知し、

前記ライブラリは、アプリケーションに資源を提供する際、前記アプリケーション特定手段によって特定されたアプリケーションに対応するアプリケーション情報管理手段の資源回収通知に前記通知手段を登録し、前記通知手段に提供した資源を対応付け、前記資源回収通知手段から資源回収の通知を受けた際、通知に使用された通知手段に対応付けられた資源だけを回収することを特徴とする請求項１記載のアプリケーション実行装置。

【請求項５】　前記資源回収通知手段に登録した前記通知手段を削除する通知手段削除手段を備えることを特徴とする請求項１記載のアプリケーション実行装置。

【請求項６】　前記資源回収通知手段から資源回収対象アプリケーションとして通知されたアプリケーションの状態を通知するアプリケーション状態通知手段を備え、

前記ライブラリは、前記アプリケーション状態通知手段から得られたアプリケーションの状態に応じて資源回収を行うことを特徴とする請求項２記載のアプリケーション実行装置。

【請求項７】　前記アプリケーション情報管理手段は、管理するアプリケーションの状態を通知するアプリケーション状態通知手段を備え、

前記ライブラリは、前記アプリケーション状態通知手段から得られたアプリケーションの状態に応じて資源回収を行うことを特徴とする請求項４記載のアプリケーション実行装置。

【請求項８】　コンピュータ読み取り可能な記録媒体であって、

アプリケーションを記憶するアプリケーション記憶手段を備える装置において、

前記アプリケーション記憶手段が記憶するアプリケーションを実行するアプリケーション実行手段と、

アプリケーションが使用する資源を回収すべきタイミングを通知手段と、

前記通知手段を登録され、前記通知手段を用いて資源回収のタイミングを通知する資源回収通知手段と、

システムが備えるデバイスを制御し、アプリケーション実行手段が実行するアプリケーションに資源を提供し、前記通知手段を前記資源回収通知手段に登録し、前記資源回収通知手段より資源回収のタイミングを通知され、資源を回収するライブラリの各機能を発揮するプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、任意のアプリケーションを実行するアプリケーション実行装置において、アプリケーションが終了時に解放しなかった資源を回収する機構に関する。

【０００２】

【従来の技術】

従来のアプリケーション実行装置における資源回収方法は、特開平８－１２３７００「資源管理方法」に記述されている。アプリケーションはタスクに割り当てられ、タスクは複数のスレッドを有しアプリケーションを実行する。アプリケーションが使用する資源は、スレッドに関するものはスレッド単位で管理され、タスクに関する物はタスク単位で管理され、システム全体に関する物は、システム全体で管理される。図１２は、これら管理状態を図示したものである。空間１２０１はシステム全体を表し、システム全体に関する資源は空間資源１２０４で管理される。システム中には複数のタスク１２０２が存在し、タスクに関する資源は、タスク資源１２０５で管理される。タスク１２０２は複数のスレッド１２０３を有し、スレッドに関する資源はスレッド資源１２０６で管理される。管理されている資源は、スレッド終了時にはスレッドに関する資源だけが回収され、タスク終了時にはタスクに関する資源とその中に含まれるスレッドに関する資源が回収さる。

【０００３】

【発明が解決しようとする課題】

しかしながら、従来の資源回収方式では、システムが使用する資源に関して、いつ回収すべきものなのか、どのような手順で回収すべきなのかを知っていなければならない。このため、システムに新規のデバイスなどを追加する場合、資源を管理する方法をシステムに組み込むため、システム自体の修正・改善・追加をしなければならない。本発明は、追加される資源の回収時期や回収方法をシステム本体に追加する必要がない、資源回収機能を持つ、アプリケーション実行装置を提供する。

【０００４】

【課題を解決するための手段】

上記課題を解決するため本発明は、アプリケーションを記憶するアプリケーション記憶手段と、前記アプリケーション記憶手段が記憶するアプリケーションを実行するアプリケーション実行手段と、アプリケーションが使用する資源の回収の指示に使用される通知手段と、前記通知手段を登録され、前記通知手段を用いて資源回収のタイミングを通知する資源回収通知手段と、システムが備えるデバイスを制御し、アプリケーション実行手段が実行するアプリケーションに資源を提供し、前記通知手段を前記資源回収通知手段に登録し、前記資源回収通知手段より資源回収のタイミングを通知され、資源を回収するライブラリを備えることとしている。

【０００５】

【発明の実施の形態】

以下、本発明に係るアプリケーション実行装置の具体的な実施の形態を、図面を参照しながら説明する。

【０００６】

（実施の形態１）

図１は、本発明に係るアプリケーション実行装置の一般的な形態を表したものであり、アプリケーション入力部１０１、アプリケーション記憶部１０２、アプリケーション実行部１０３、ライブラリー格納部１０４、デバイス部１０５、制御部１０６で構成される。

【０００７】

アプリケーション入力部１０１は、実行すべきアプリケーションを受け付ける。具体的にはフロッピーディスクやＣＤを等のメディアを通してアプリケーションが供給される場合、本構成要素はフロッピーディスクドライブ、ＣＤドライブ等で構成される。またネットワークや放送局らからの電波をを通してアプリケーションが配信される場合は、ネットワークインターフェースボードや放送受信機器で構成される。内蔵されるアプリケーションのみを実行する装置においては、アプリケーション記憶部１０２にアプリケーションを記憶しておくだけで良いので、本要素は不要となる。

【０００８】

アプリケーション記憶部１０２は、実行すべきアプリケーションを記憶する。具体的には、ＲＡＭ、ＲＯＭ、ハードディスクなどの内蔵機器や、ＣＤドライブ、フロッピーディスクや、フロッピーディスクなどのリムーバブルメディア等で構成される。

【０００９】

アプリケーション実行部１０３は、アプリケーション記憶部１０２が記憶するアプリケーションを実行する。内部には、逐次実行部１０３ａ、資源回収通知部１０３ｂ、アプリケーション識別部１０３ｃを構成要素として持つ。

【００１０】

逐次実行部１０３ａはアプリケーション記憶部１０２が記憶するアプリケーションを実行する。アプリケーションがバイナリープログラムの場合、逐次実行部１０３ａは、バイナリープログラムをそのまま実行するＣＰＵ等で構成される。また、アプリケーションがＪａｖａバイトコードのような中間コードの場合はＣＰＵ及びその上で実行されるインタープリタやバーチャルマシーンとよばれる中間コードを逐次解析し実行するモジュール等で構成される。アプリケーションは、システムが備えているデバイスを制御するため、デバイスを制御するためのライブラリを呼び出す。逐次実行部１０３ａは、アプリケーションがライブラリを呼び出すとき、ライブラリ格納部１０４が格納するライブラリから必要なものを呼び出し実行する。

【００１１】

資源回収通知部１０３ｂは、ライブラリ格納部１０４が格納するライブラリから、資源回収タイミングの通知するためのコールバック関数を受け付ける。ライブラリは規定されたインターフェースを持つコールバック関数を資源回収通知部１０３ｂに登録する。資源回収通知部１０３ｂは、資源回収を行うタイミングで登録されたコールバック関数を呼び出す。この際、資源回収対象のアプリケーションの識別子も引き渡す。

【００１２】

アプリケーション識別部１０３ｃは、ライブラリがアプリケーションから呼び出された際、呼び出したアプリケーションを特定するための識別子を提供する。ライブラリがアプリケーションから呼び出された際、アプリケーション識別部１０３ｃを呼び出すと、アプリケーションを特定する識別子を返してくれる。

【００１３】

ライブラリ格納１０４は、システムが備えるデバイスを制御するためのライブラリを格納し、具体的にはＲＯＭやハードディスクなどで構成される。ライブラリ格納部１０４は、一般に複数のライブラリ１０４ａ、１０４ｂ、、、１０４ｎを格納し、各ライブラリーはデバイス部１０５に属するデバイス１０５ａ、１０５ｂ、、、１０５ｎをプリケーションが制御するための関数群で構成される。また、各ライブラリは資源回収通知部１０３ｂに登録するためのコールバック関数を備え、ライブラリが制御するデバイスがアプリケーションによって確保され使用されている状態を保持する。各ライブラリはアプリケーションによってデバイスを確保し制御する際に、コールバック関数を資源回収通知部１０３ｂに登録する。また、アプリケーション識別部１０３ｃを呼び出し、アプリケーション識別子を獲得し、確保した資源と組にして保持しておく。

【００１４】

アプリケーションが異常終了などにより、ライブラリを通して確保した資源を解放しなかった場合、予め登録したコールバック関数が、資源回収通知部１０３ｂより資源回収のタイミングで呼び出される。この時に、アプリケーション識別子が渡されるので、保持しているアプリケーション識別子と資源との組を参照し

、渡されたアプリケーション識別子に対応する資源が解放されてなければ、この資源を回収し、他のアプリケーションが使用できるようにする。

【００１５】

ファイルシステムを制御するファイルシステムライブラリが存在する場合を想定し、具体的に説明する。図２は、ファイルシステムライブラリが保持するアプリケーション識別子と資源を組にした情報を保持するテーブルデータである。（１）は、アプリケーション識別子「１」を持つアプリケーションがファイル「ａ．ｔｘｔ」を使用している状態を表す。この時、既にファイルシステムライブラリはコールバック関数を資源回収通知部１０３ｂに登録している。

【００１６】

新しく「ｂ．ｔｘｔ」を使用するためファイルシステムライブラリの関数が呼び出されたとする。ファイルシステムライブラリはコールバック関数は既に登録しているので、再度登録することは行わない。ファイルシステムライブラリはアプリケーション識別部１０３ｃを呼び出し、アプリケーション識別子「２」を得る。ファイルシステムライブラリは、ファイル「ｂ．ｔｘｔ」と獲得したアプリケーション識別子を組として保持しておく。その結果、テーブルデータは図２（２）のように更新される。このあと、確保したファイルをアプリケーションが制御できるようにする。

【００１７】

アプリケーション識別子「２」を持つアプリケーションが異常終了し、ファイル資源を解放せずに終了したとする。資源回収通知部１０３ｂは、アプリケーション識別子「２」が確保した資源を回収するために、ファイルシステムライブラリが登録したコールバック関数を呼び出す。ファイルシステムライブラリは図２（２）の状態のテーブルデータを参照し、ファイル「ｂ．ｔｘｔ」を解放し、テーブルデータを図２（１）の状態に戻す。

【００１８】

デバイス部１０５はシステムが備える全デバイスの集合である。具体的には、メモリーデバイス、ディスプレイデバイス、入力デバイス（キーボード、リモコン等）、ファイルシステム、ネットワークデバイスなどが含まれ、それぞれのデ

バイスはライブラリ格納部１０６が格納するライブラリによってアプリケーションから制御可能である。

【００１９】

制御部１０６は、システム上で動作するアプリケーションの開始、停止、再開、終了などを指示する。

【００２０】

アプリケーションの開始は、アプリケーション入力部１０１に、アプリケーションを受け付け、アプリケーション記憶部１０２に記憶する指示を行い、その後、逐次実行部１０３ａにアプリケーションを実行する指示を出す。

【００２１】

アプリケーションの停止は、逐次実行部１０３ａに実行しているアプリケーションの中断を指示する。また、場合によっては資源回収通知部１０３ｂに中断したアプリケーションの資源回収を指示する。この際、資源回収通知部１０３ｂは登録されたコールバック関数を呼び出し、停止されたアプリケーションのアプリケーション識別子を引き渡す。

【００２２】

アプリケーションの再開では、逐次実行部１０３ａに停止しているアプリケーションを実行する指示を出す。

【００２３】

アプリケーションの終了は、逐次実行部１０３ａに実行しているアプリケーションの終了を指示する。また、資源回収通知部１０３ｂに終了したアプリケーションの資源回収を指示する。この際、資源回収通知部１０３ｂは登録されたコールバック関数を呼び出し、終了されたアプリケーションのアプリケーション識別子を引き渡す。

【００２４】

図３は、本実施の形態のアプリケーション実行装置において、ライブラリが呼び出され資源が獲得される動作を説明するフローチャートである。

【００２５】

ライブラリがアプリケーションから資源を確保する指示を受けると、まず要求

された資源を確保する（ステップＳ３０１）。次に、コールバック関数を既に登録しているかどうかを判断し（ステップＳ３０２）、登録していなければ資源回収通知部１０３ｂにコールバック関数を登録する（ステップＳ３０３）。アプリケーション識別部１０３ｃを呼び出し、アプリケーション識別子を取得し（ステップＳ３０４）、確保した資源とアプリケーション識別子を組として保持しておく（ステップＳ３０５）。これらの資源管理処理を行った後、確保した資源をアプリケーションに提供する（ステップＳ３０６）。

【００２６】

図４は、本実施の形態のアプリケーション実行装置において、ライブラリが登録したコールバック関数が呼び出され資源が回収される動作を説明するフローチャートである。

【００２７】

コールバック関数が呼び出されると、ライブラリは資源を回収するアプリケーションを特定するアプリケーション識別子を取得する（ステップＳ４０１）。次に、確保している資源とアプリケーション識別子の組を１つ取り出す（ステップＳ４０２）。取得したアプリケーション識別子と、確保した資源と組になっているアプリケーション識別子を比較し（ステップＳ４０３）、一致した場合は、アプリケーション識別子と組で保持されている資源を解放し（ステップＳ４０４）、確保している資源とアプリケーション識別子の組を削除する（ステップＳ４０５）。ステップＳ４０２からステップＳ４０５までを全ての確保している資源とアプリケーション識別子の組に対して行う（ステップＳ４０６）。

【００２８】

（実施の形態２）

図５は、本発明に係るアプリケーション実行装置の一例としてＪａｖａアプリケーションを実行するシステムの形態を表したものであり、アプリケーション入力部１０１、アプリケーション記憶部１０２、アプリケーション実行部５０３、ライブラリー格納部５０４、デバイス部１０５、制御部１０６で構成される。第１の実施例と同じ構成要素は同じ符号を割り当て説明は省略する。

【００２９】

アプリケーション実行部５０３は、アプリケーション記憶部１０２が記憶するＪａｖａアプリケーションを実行する。内部には、バーチャルマシーン部５０３ａ、アプリケーション管理部５０３ｂを構成要素として持つ。

【００３０】

バーチャルマシーン部５０３ａはアプリケーション記憶部１０２が記憶するＪａｖａアプリケーションのバイトコードを実行する。Ｊａｖａアプリケーションは、システムが備えているデバイスを制御するため、デバイスを制御するためのクラスライブラリを呼び出す。バーしゃんるマシーン部５０３ａは、ｊａｖａアプリケーションがクラスライブラリを呼び出すとき、ライブラリ格納部５０４が格納するクラスライブラリから必要なクラスを呼び出し実行する。

【００３１】

アプリケーション管理部５０３ｂは、実行されているＪａｖａアプリケーションを管理する。アプリケーション情報を格納するためのクラスからインスタンスを生成し、そのインスタンスにアプリケーションの情報を格納する。５０３ｃは、アプリケーション情報を格納するインスタンスである。ここではこれをアプリケーション情報インスタンスと呼ぶ。

【００３２】

アプリケーション管理部５０３ｂは、第１の実施の形態におけるアプリケーション識別部１０３ｃの役割を備える。つまり、ライブラリがアプリケーションから呼び出された際、呼び出したアプリケーションを特定するための識別子を提供する。ここで、アプリケーション管理部５０３ｂは、識別子としてアプリケーション情報インスタンスを引き渡す。

【００３３】

アプリケーション情報インスタンスは、第１の実施例における資源回収通知部１０３ｂの役割を備える。つまり、規定されたコールバック関数を受け付け、資源回収時に登録されたコールバック関数を呼び出す。一般にｊａｖａ言語では、コールバック関数の登録を、コールバック関数に相当するメソッドを備えるクラスのインスタンスの登録として表現する。アプリケーション情報インスタンスに登録するクラスの形式を規定するため、Ｊａｖａ言語におけるインターフェース

を規定する。図６は、このインターフェースの一例である。このｒｅｓｏｕｒｃｅＣｏｌｌｅｃｔｉｏｎＬｉｓｔｅｎｅｒインターフェースは、ｕｐｄａｔｅメソッドを規定している。各ライブラリは、このインターフェースをｉｍｐｌｅｍｅｎｔｓし、ｕｐｄａｔｅメソッドに資源回収処理を記述したクラスを用意し、インスタンスを作成してアプリケーション情報インスタンスに登録する。アプリケーション情報インスタンスは、資源回収を行うタイミングで、登録されたインスタンスのｕｐｄａｔｅメソッドを呼び出す。ここで、アプリケーション情報インスタンスに登録する資源回収用のインスタンスを資源回収インスタンスと呼ぶ。

【００３４】

第１の実施の形態では、ライブラリは資源回収用のコールバック関数をアプリケーションに依存せず、資源回収通知部１０３ｂに登録したが、本第２の実施の形態では、資源を使用しているアプリケーションを管理しているアプリケーション情報インスタンスに資源回収インスタンスを登録する。アプリケーション情報インスタンスは、管理しているアプリケーションから資源を回収するタイミングで資源回収インスタンスのメソッドを呼び出す。この結果、ライブラリーは使用されるアプリケーションの数だけ、資源回収インスタンスを作成し登録しなければならないが、資源回収処理は効率的である。第１の実施の形態では、一部のライブラリしか使用していないアプリケーションが停止・終了した場合にも、全てのライブラリからのコールバック関数を呼び出す必要があった。しかし、本第２の実施の形態では、資源回収インスタンスはアプリケーションと関連付けて登録されているので、停止・終了等したアプリケーションに登録された資源回収インスタンスのみが呼び出される。また、呼び出されること自体で、資源を回収すべきアプリケーションを特定できているので、資源回収インスタンスのメソッド呼び出し時にアプリケーション識別子を引き渡す必要がない。

【００３５】

更に、アプリケーション情報インスタンスには、登録した資源回収インスタンスを削除するメソッドを用意しても良い。これはアプリケーションが実行中に資源を解放し、ライブラリが資源回収の通知を必要としなくなった時に、この削除

用メソッドを呼び出しアプリケーション情報インスタンスから資源回収インスタンスを削除することで、余分な資源回収インスタンスのメソッド呼び出しを減らすことが出来る。

【００３６】

また、回収資源は、回収するタイミングで異なるかもしれない。具体的には、アプリケーションが停止した状態で回収してもよい資源、回収してはいけない資源などがあるかもしれない。これを知るために、アプリケーション情報インスタンスにアプリケーションの状態を知るためのメソッドを用意してもよい。ライブラリーは、資源回収インスタンスのメソッドが呼び出された際、アプリケーションの状態を知るためのメソッドを呼び出し、その結果に応じて処理を行えばよい。

【００３７】

図７は、上記の機能を含むアプリケーション情報インスタンスを作るためのクラスの一例である。ａｐｐｌｉｃａｔｉｏｎＰｒｏｘｙは、資源回収インスタンスを受け付けるａｄｄＬｉｓｔｅｎｅｒメソッドを用意し、またこれを削除するｒｅｍｏｖｅＬｉｓｔｅｎｅｒメソッドも用意している。アプリケーションの状態を知るためにｇｅｔＳｔａｔｕｓメソッドを用意している。この例ではアプリケーションの情報としてアプリケーション名を保持するだけであるが、その他の情報を保持していても実施可能である。

【００３８】

ライブラリ格納５０４は、システムが備えるデバイスを制御するためのｊａｖａクラスライブラリを格納し、具体的にはＲＯＭやハードディスクなどで構成される。ライブラリ格納部５０４は、一般に複数のライブラリ５０４ａ、５０４ｂ、、、５０４ｎを格納し、各ライブラリーはデバイス部１０５に属するデバイス１０５ａ、１０５ｂ、、、１０５ｎをアプリケーションが制御するためのクラス群で構成される。また、各ｊａｖａクラスライブラリはアプリケーション情報インスタンス５０３ｃに登録するための資源回収クラスを備え、ｊａｖａクラスライブラリが制御するデバイスがアプリケーションによって確保され使用されている状態を保持する。各ｊａｖａクラスライブラリはアプリケーションによってデ

バイスを確保し制御する際に、アプリケーション管理部１０３ｂを呼び出し、アプリケーション情報インスタンスを取得し、資源回収インスタンスを資源回収クラスから生成し、アプリケーションに対応するアプリケーション情報インスタンス５０３ｃに登録する。また、確保した資源は資源回収インスタンスに対応付けて記録しておく。

【００３９】

アプリケーションが異常終了などにより、ｊａｖａクラスライブラリを通して確保した資源を解放しなかった場合、予め登録した資源回収インスタンスのメソッドが、アプリケーション情報インスタンスから呼び出される。ｊａｖａクラスライブラリは、呼び出された資源回収インスタンスに対応付けられている資源を解放する。

【００４０】

ファイルシステムを制御するファイルシステムクラスライブラリが存在する場合を想定し、具体的に説明する。図８は、ファイルシステムライブラリが用意する資源回収クラスである。資源回収クラスから生成される資源回収インスタンスは、このインスタンスが登録されるアプリケーション情報インスタンスを内部変数ａｐｐと対応する資源を内部変数ｎａｍｅに保存する。メソッドｕｐｄａｔｅは、資源回収時に呼び出されるメソッドである。

【００４１】

図９は、ファイルシステムクラスライブラリが保持する資源回収インスタンスを表している。９０１は資源回収インスタンスを保持するインスタンステーブルである。（１）は、アプリケーション情報インスタンス（ここでは仮に「１」と表現する）「１」で管理されるアプリケーションがファイル「ａ．ｔｘｔ」を使用している状態を表す。この時、資源回収インスタンス９０２はアプリケーション情報インスタンス「１」に既に登録されている。

【００４２】

新しく「ｂ．ｔｘｔ」を使用するためファイルシステムライブラリの関数が呼び出されたとする。ファイルシステムクラスライブラリは、アプリケーション管理部５０４ｂを呼び出し、アプリケーションに対応するアプリケーション情報イ

ンスタンス「２」を取得する。このアプリケーション情報インスタンスと既登録の資源回収インスタンス中のアプリケーション情報インスタンスを比較する。一致したものが無いので、新規の資源回収インスタンスを生成し、獲得したアプリケーション情報インスタンスに登録する。また、資源回収インスタンスには確保した資源「ｂ．ｔｘｔ」とアプリケーション情報インスタンス「２」を記録する。図９（２）この状態を表す。

【００４３】

アプリケーション情報インスタンス「２」が管理するアプリケーションが異常終了し、ファイル資源を解放せずに終了したとする。アプリケーション情報インスタンス「２」は、登録された資源回収インスタンスのメソッドを呼び出す。ファイルシステムクラスライブラリは資源回収インスタンスに記録された資源、ファイル「ｂ．ｔｘｔ」を解放すし、この資源回収インスタンスも不要なので解放する。必要ならば、アプリケーション情報インスタンスの資源回収インスタンスの削除メソッドも呼び出し、再び呼び出されないようにする。

【００４４】

図１０は、本実施の形態のアプリケーション実行装置において、ｊａｖａクラスライブラリが呼び出され資源が獲得される動作を説明するフローチャートである。ｊａｖａクラスライブラリは資源を要求されると、資源を確保する（ステップＳ１００１）。次に、アプリケーション管理部５０３ｂから資源要求してきたアプリケーションに対応するアプリケーション情報インスタンスを獲得する（ステップＳ１００２）。ｊａｖａクラスライブラリは、保持している資源回収インスタンスの１つを取り出し（ステップＳ１００３）、アプリケーション管理部５０３ｂから取得したアプリケーション情報インスタンスと資源回収インスタンス中のアプリケーション情報インスタンスを比較する（ステップＳ１００４）。一致すれば、一致したアプリケーション情報インスタンスを含む資源回収インスタンスに確保した資源を記録し（ステップＳ１００５）、確保した資源をアプリケーションに提供する（ステップＳ１００９）。全資源回収インスタンスに対してアプリケーション情報インスタンスの比較を行い（ステップＳ１００６）、一致しなければ、新規に資源回収インスタンスを生成し、確保した資源と、アプリケ

ーション管理部５０３ｂから取得したアプリケーション情報インスタンスを記録する（ステップＳ１００７）。次に作成した資源回収インスタンスを、アプリケーション情報インスタンスに登録する（ステップＳ１００８）。そして、確保した資源をアプリケーションに提供する（ステップＳ１００９）。

【００４５】

図１１は、本実施の形態のアプリケーション実行装置において、ｊａｖａクラスライブラリが登録した資源回収インスタンスのメソッドが呼び出され資源が回収される動作を説明するフローチャートである。資源回収インスタンスのメソッドが呼び出されると、資源回収インスタンスに記録されているアプリケーション情報インスタンスのアプリケーション状態取得メソッドを呼び出し、アプリケーションが停止されたのか、終了されたのかなどの情報を得る（ステップＳ１１０１）。その状態に応じて、資源回収すべきかどうかを判定する（ステップＳ１１０２）。回収する場合は、資源回収インスタンスに記録されている資源を解放し（ステップＳ１１０３）、資源回収インスタンス自体を解放する（ステップＳ１１０４）。

【００４６】

尚、実施の形態１において、資源回収時にコールバック関数に通知されるアプリケーションは１つだけであった。しかし、複数のアプリケーションが同時に終了する場合には、同時に複数のアプリケーションの識別子をコールバック関数に通知することが可能である。ライブラリは複数のアプリケーションの終了通知を一度に受け取ることによって、より効果的な資源回収を行うことが可能な場合がありえる。例えば、あるライブラリが管理する資源が複数のアプリケーションを終了することによって全て開放される場合、管理情報の一部を修正するのではなく、管理テーブルを初期化するだけで良い場合がある。この場合、より高速に資源回収を行うことが出来る。

【００４７】

尚、実施の形態２において、アプリケーションの状態に応じて、特定の一部の資源のみを解放することも可能である。この場合、資源回収インスタンスの解放は、全資源が解放された時に行う。

【００４８】

【発明の効果】

以上説明したように、本発明によれば、アプリケーションを記憶するアプリケーション記憶手段と、前記アプリケーション記憶手段が記憶するアプリケーションを実行するアプリケーション実行手段と、アプリケーションが使用する資源の回収の指示に使用される通知手段と、前記通知手段を登録され、前記通知手段を用いて資源回収のタイミングを通知する資源回収通知手段と、システムが備えるデバイスを制御し、アプリケーション実行手段が実行するアプリケーションに資源を提供し、前記通知手段を前記資源回収通知手段に登録し、前記資源回収通知手段より資源回収のタイミングを通知され、資源を回収するライブラリを備えることとしている。

【００４９】

ライブラリは規定された通知手段を登録し、資源回収通知手段は通知手段を通じてライブラリに資源回収の指示を出すことによって、資源回収操作自体をライブラリの中に実現することができるため、システム本体の修正無しに、ライブラリの削除・登録が可能となり、システムの保守性が高まる。

【００５０】

更に、問い合わせに応じて前記アプリケーション実行手段が実行するアプリケーションを特定するアプリケーション特定手段を備え、前記資源回収通知手段は、資源回収通知時に資源回収の対象であるアプリケーションを通知し、前記ライブラリは、アプリケーションに提供する資源を前記アプリケーション特定手段によって特定されたアプリケーションと対応付けて保持し、前記資源回収通知手段から資源回収の通知を受けた際、同時に通知される資源回収対象のアプリケーションに提供した資源だけを回収することとしている。

【００５１】

複数のアプリケーションが実行されるシステムにおいて、特定のアプリケーションのみが使用する資源だけを解放することができる。

【００５２】

更に、前記資源回収通知手段は、資源回収通知時に資源回収の対象であるアプ

リケーションが複数あるとき、これら複数のアプリケーションを同時に通知することとしている。

【００５３】

ライブラリーが管理する資源によっては複数の資源を同時に解放できる場合もある。資源回収すべき複数のアプリケーションをライブラリーが同時に受け取ることにより、呼び出しによるオーバーヘッドを削減し、かつライブラリが管理する資源の特性を生かした効率的な資源回収を実現することができる。

【００５４】

更に、アプリケーション毎に、前記資源回収通知手段を備えるアプリケーション情報管理手段と、問い合わせに応じて前記アプリケーション実行手段が実行するアプリケーションを特定するアプリケーション特定手段を備え、前記アプリケーション情報手段が備える前記資源回収通知手段は、前記アプリケーション情報管理手段が管理するアプリケーションから資源回収を行う場合だけ、登録されている通知手段を用いて資源回収のタイミングを通知し、前記ライブラリは、アプリケーションに資源を提供する際、前記アプリケーション特定手段によって特定されたアプリケーションに対応するアプリケーション情報管理手段の資源回収通知に前記通知手段を登録し、前記通知手段に提供した資源を対応付け、前記資源回収通知手段から資源回収の通知を受けた際、通知に使用された通知手段に対応付けられた資源だけを回収する構成とすることとしている。

【００５５】

アプリケーションに対応してライブラリが通知手段を登録するので、資源通知手段はアプリケーションに関係するライブラリだけを呼び出すことができるので、余分な呼び出し回数を削減することが出来る。

【００５６】

更に、前記資源回収通知手段に登録した前記通知手段を削除する通知手段削除手段を備えることとしている。

【００５７】

アプリケーションが実行中に資源を解放し、ライブラリが資源回収の通知を必要としなくなった時、通知手段削除手段を用いて削除することにより、資源回収

時に余分な呼び出し回数を削減することが出来る。

【００５８】

更に、前記資源回収通知手段から資源回収対象アプリケーションとして通知されたアプリケーションの状態を通知するアプリケーション状態通知手段を備え、前記ライブラリは、前記アプリケーション状態通知手段から得られたアプリケーションの状態に応じて資源回収を行うこととしている。

【００５９】

ライブラリは、アプリケーションの状態に応じてアプリケーションが使用している資源を回収することが出来る。例えば、非表示状態のアプリケーションが使用しているグラフィックスメモリーなどを回収することなども可能である。

【００６０】

更に、前記アプリケーション情報管理手段は、管理するアプリケーションの状態を通知するアプリケーション状態通知手段を備え、前記ライブラリは、前記アプリケーション状態通知手段から得られたアプリケーションの状態に応じて資源回収を行う。

【００６１】

ライブラリは、アプリケーションの状態に応じてアプリケーションが使用している資源を回収することが出来る。例えば、非表示状態のアプリケーションが使用しているグラフィックスメモリーなどを回収することなども可能である。

【００６２】

更に、コンピュータ読み取り可能な記録媒体であって、アプリケーションを記憶するアプリケーション記憶手段を備える装置において、前記アプリケーション記憶手段が記憶するアプリケーションを実行するアプリケーション実行手段と、アプリケーションが使用する資源を回収すべきタイミングを通知手段と、前記通知手段を登録され、前記通知手段を用いて資源回収のタイミングを通知する資源回収通知手段と、システムが備えるデバイスを制御し、アプリケーション実行手段が実行するアプリケーションに資源を提供し、前記通知手段を前記資源回収通知手段に登録し、前記資源回収通知手段より資源回収のタイミングを通知され、資源を回収するライブラリの各機能を発揮するプログラムとすることとしている

。このような構成によって、資源回収機構を有するアプリケーション実行装置としてコンピュータを利用することができる。

【図面の簡単な説明】

【図１】

本発明に係る資源回収機能を有するアプリケーション実行装置の実施の形態１の構成図

【図２】

（１）実施の形態１に含まれるライブラリの一例であるファイルシステムライブラリが、アプリケーションとファイル資源を対応付けて保持している状態を示す図

（２）実施の形態１に含まれるライブラリの一例であるファイルシステムライブラリが、アプリケーションとファイル資源を対応付けて保持している状態を示す図

【図３】

実施の形態１において、資源提供の際の動作を表すフローチャート

【図４】

実施の形態１において、資源回収の際の動作を表すフローチャート

【図５】

本発明に係る資源回収機能を有するアプリケーション実行装置の実施の形態２の構成図

【図６】

実施の形態２に含まれるアプリケーション情報インスタンスが受け付ける資源回収インスタンスを規定するインターフェースの一例を示す図

【図７】

実施の形態２に含まれるアプリケーション情報インスタンスの構成の一例を示す図

【図８】

実施の形態２に含まれるｊａｖａクラスライブラリの一例としてのファイルシステムクラスライブラリが備える資源回収インスタンスの一例を示す図

【図９】

（１）実施の形態２に含まれるライブラリの一例であるファイルシステムクラスライブラリが、資源回収インスタンスにアプリケーション情報インスタンスとファイル資源を対応付けて保持している状態を示す図

（２）実施の形態２に含まれるライブラリの一例であるファイルシステムクラスライブラリが、資源回収インスタンスにアプリケーション情報インスタンスとファイル資源を対応付けて保持している状態を示す図

【図１０】

実施の形態２において、資源提供の際の動作を表すフローチャート

【図１１】

実施の形態２において、資源回収の際の動作を表すフローチャート

【図１２】

従来のアプリケーション実行装置が管理する資源の管理方法を示す図

【符号の説明】

１０１　アプリケーション入力部

１０２　アプリケーション記憶部

１０３　アプリケーション実行部

１０３ａ　逐次実行部

１０３ｂ　資源回収通知部

１０３ｃ　アプリケーション識別部

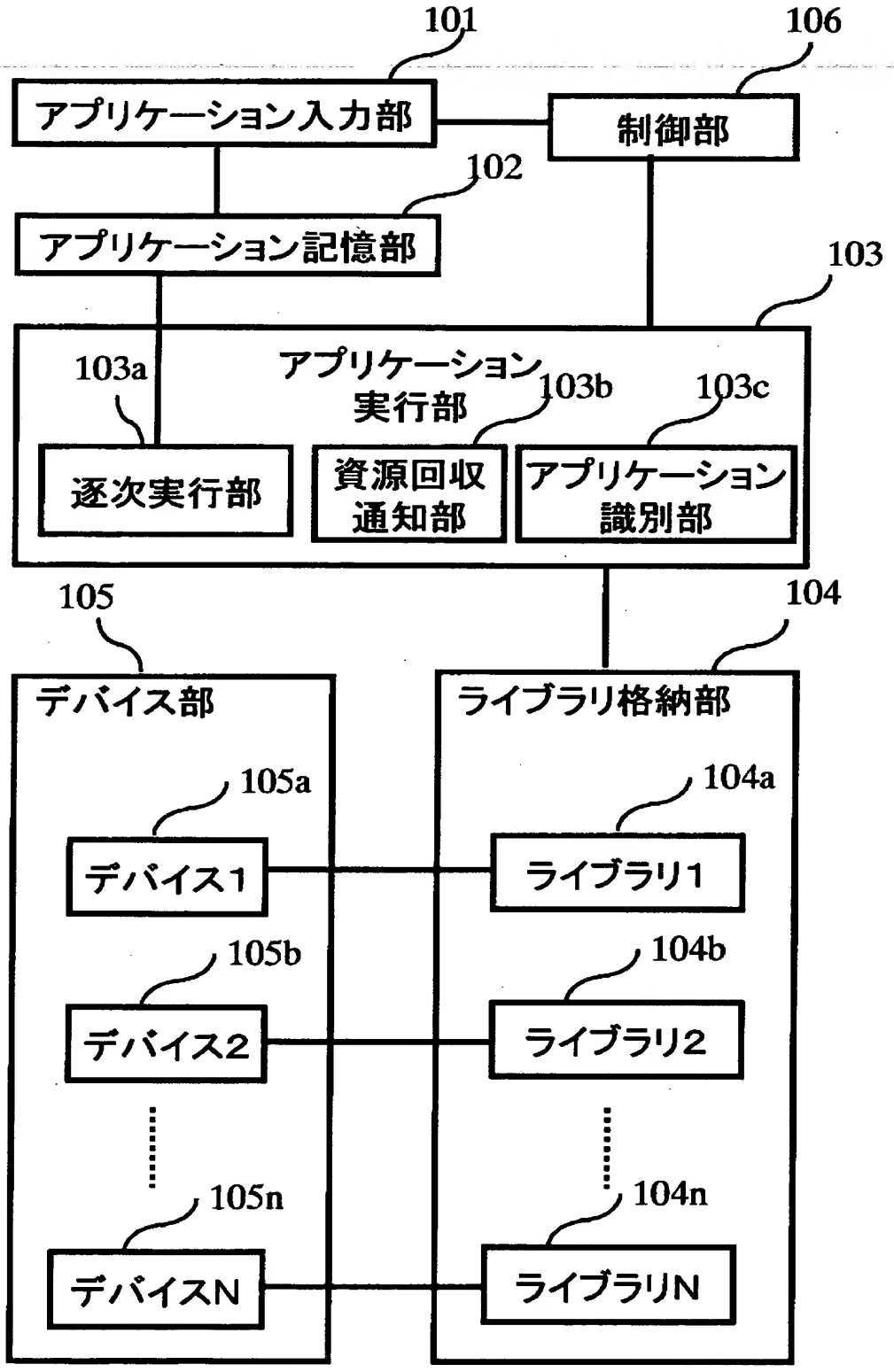
１０４　ライブラリ格納部

１０５　デバイス部

１０６　制御部

【書類名】　　図面

【図１】


101
106
アプリケーション入力部
制御部
102
アプリケーション記憶部
103
103a
アプリケーション
実行部
103b
103c
逐次実行部
資源回収
通知部
アプリケーション
識別部
105
104
デバイス部
ライブラリ格納部
105a
104a
デバイス1
ライブラリ1
105b
104b
デバイス2
ライブラリ2
105n
104n
デバイスN
ライブラリN

【図２】

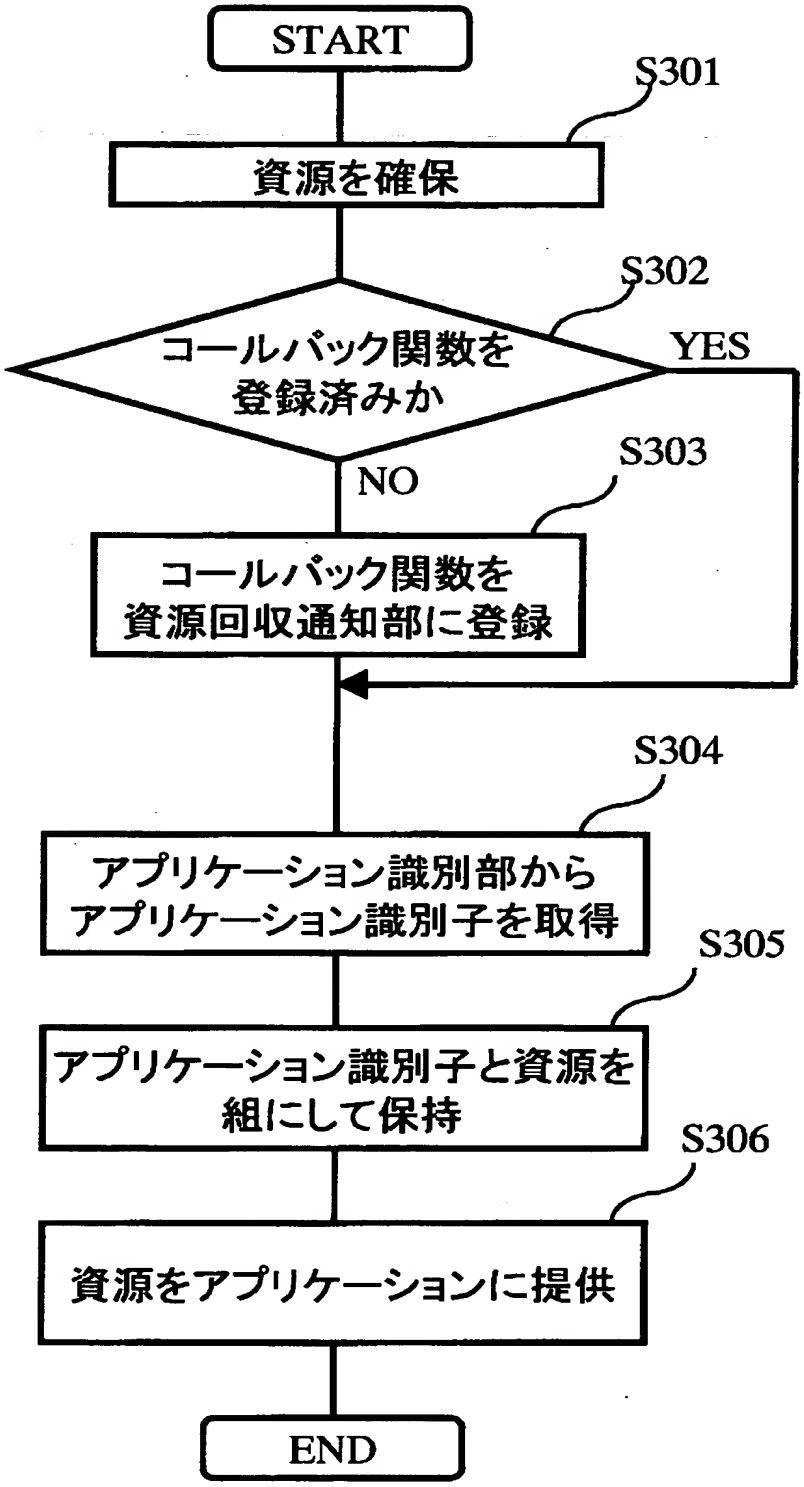
(1)

アプリケーション識別子	ファイル識別子
1	a.txt

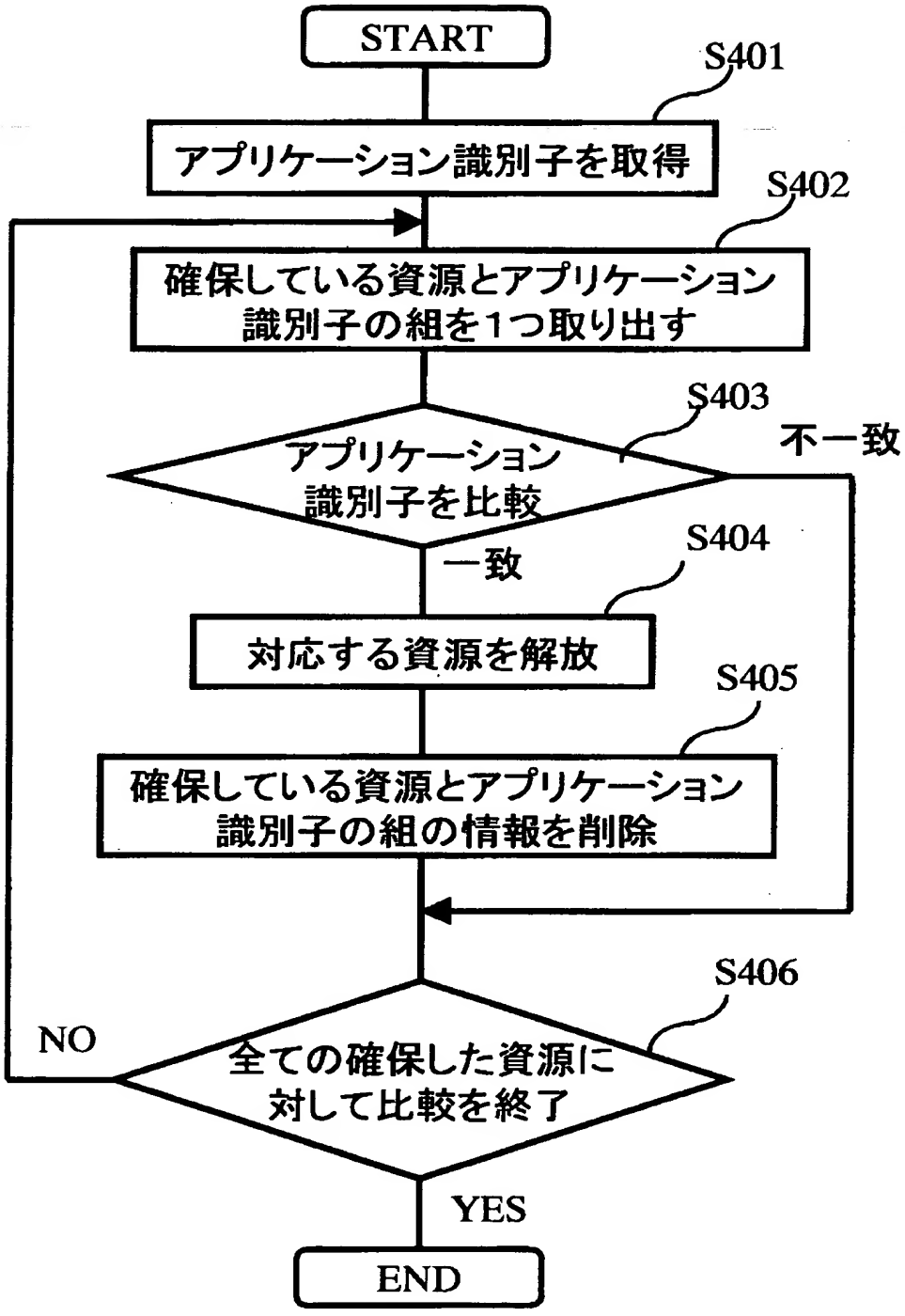
(2)

アプリケーション識別子	ファイル識別子
1	a.txt
2	b.txt

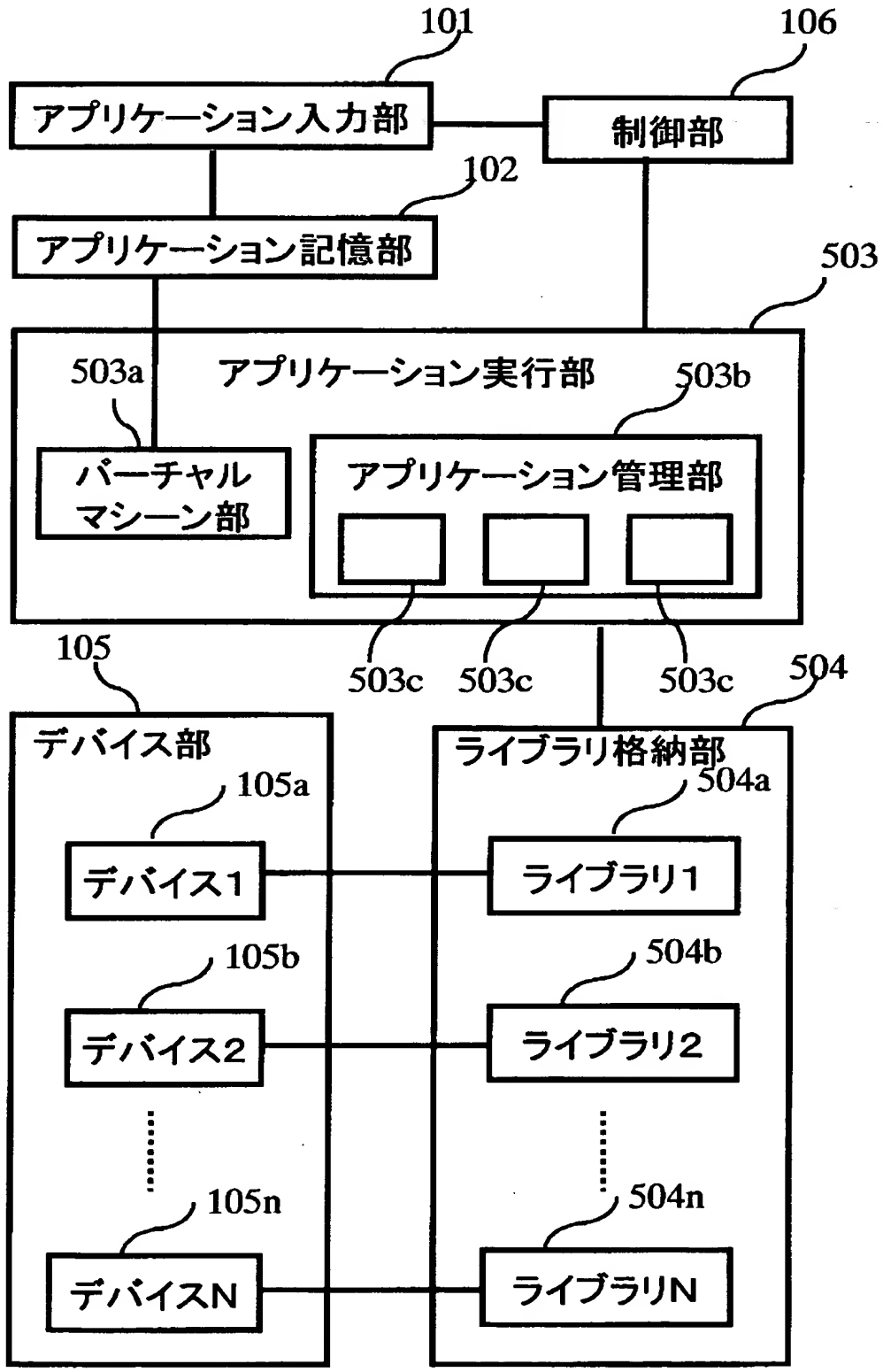
【図３】


START
S301
資源を確保
S302
コールバック関数を
登録済みか
YES
NO
S303
コールバック関数を
資源回収通知部に登録
S304
アプリケーション識別部から
アプリケーション識別子を取得
S305
アプリケーション識別子と資源を
組にして保持
S306
資源をアプリケーションに提供
END

【図４】


START
S401
アプリケーション識別子を取得
S402
確保している資源とアプリケーション識別子の組を1つ取り出す
S403
アプリケーション識別子を比較
不一致
一致
S404
対応する資源を解放
S405
確保している資源とアプリケーション識別子の組の情報を削除
S406
全ての確保した資源に対して比較を終了
NO
YES
END

【図５】


101
106
アプリケーション入力部
制御部
102
アプリケーション記憶部
503
503a
アプリケーション実行部
503b
バーチャル
マシーン部
アプリケーション管理部
105
503c
503c
503c
504
デバイス部
ライブラリ格納部
105a
504a
デバイス1
ライブラリ1
105b
504b
デバイス2
ライブラリ2
105n
504n
デバイスN
ライブラリN

【図６】

```
public interface resourceCollectionListener {
   public void update();
}
```

【図７】

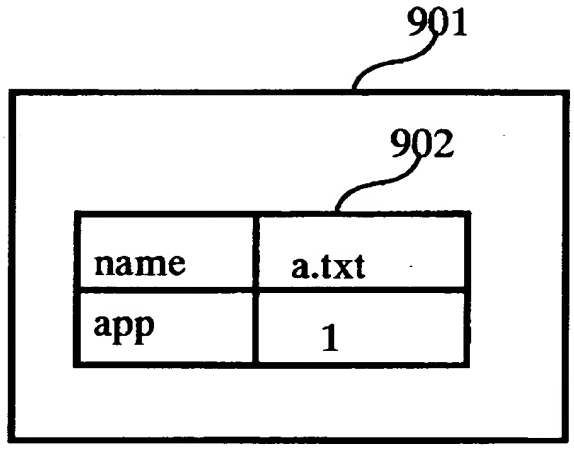
```
public class applicationProxy {
   public string name;  // application name

   public int getStatus() {
      // get the status of application, paues or destroy
   }

   public addListener(resourceCollectionListener l) {
      // register resource collection instance
   }

   public removeListener(resourceCollectionListener l) {
     // remove resource collection instance
   }
}
```

【図８】

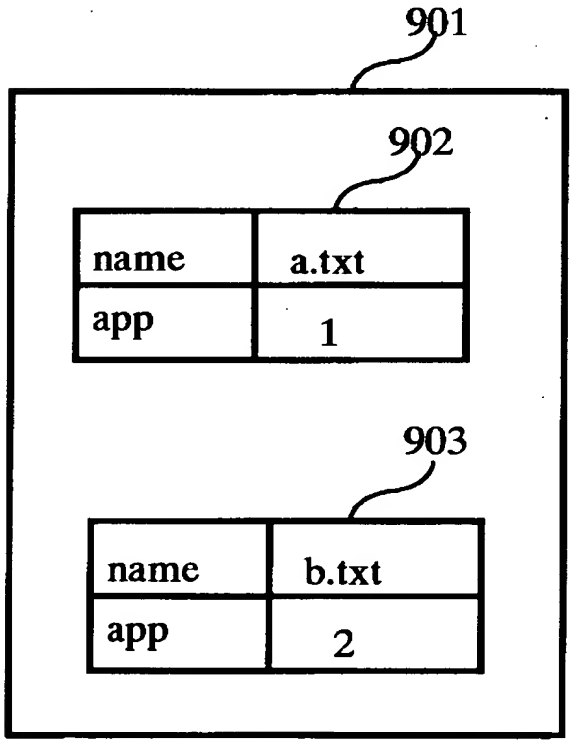
```
public class fileCollectionListener
            implements resourceCollectionListener {
   String name;  // filename
   ApplicationProxy app;
                        // Application information instance

   public void update() {
      // release file which has filename
   }
}
```

【図９】

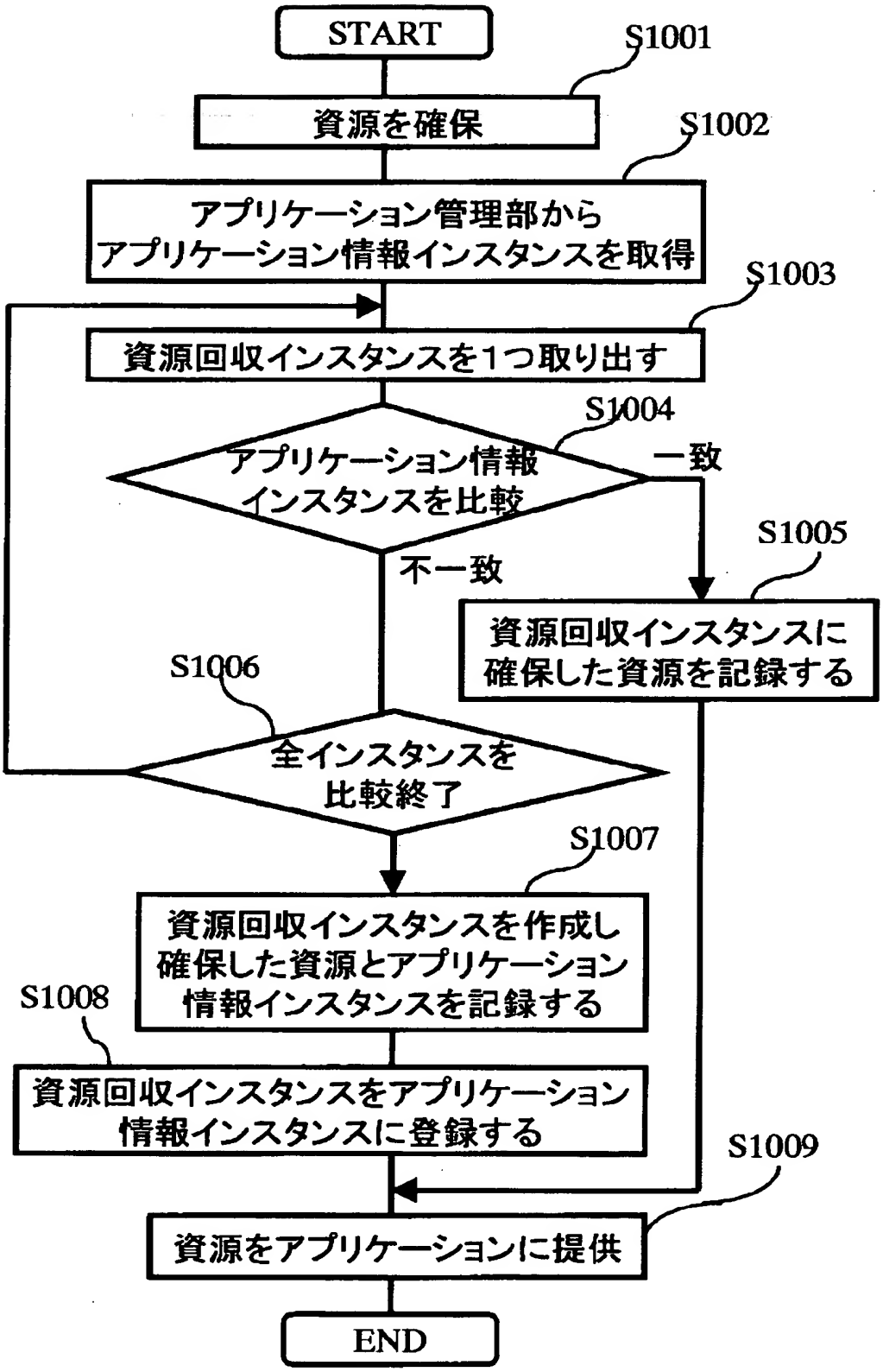
(1)


901
902
name
a.txt
app
1

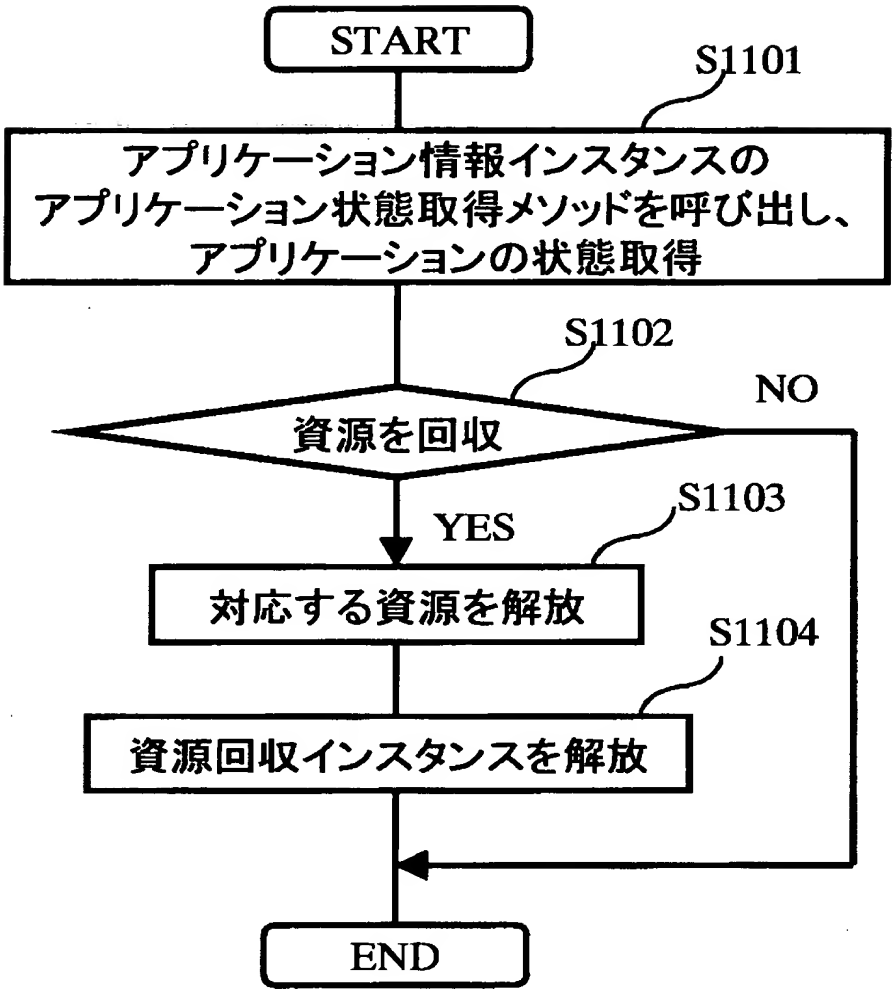
(2)


901
902
name
a.txt
app
1
903
name
b.txt
app
2

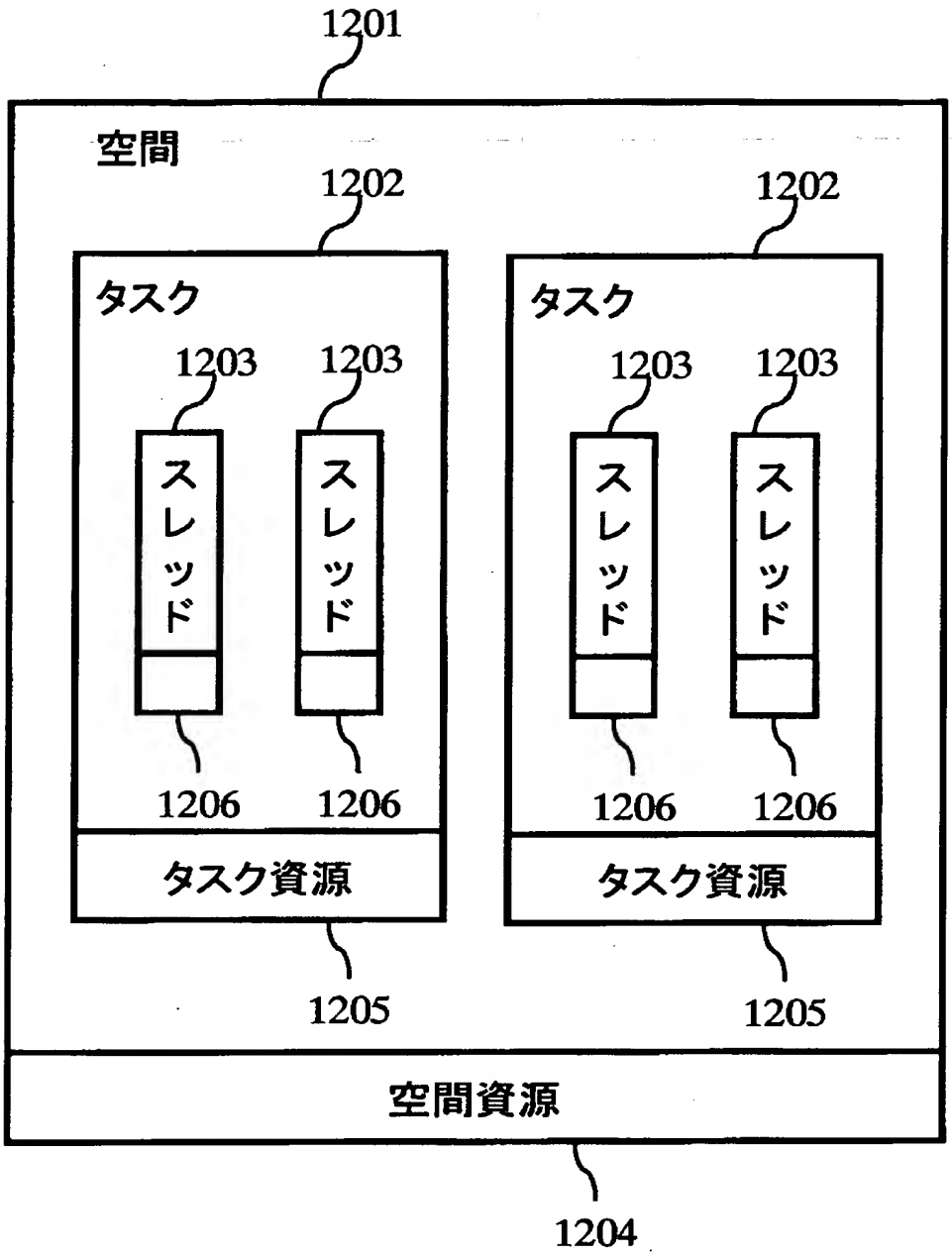
【図１０】


START
S1001
資源を確保
S1002
アプリケーション管理部から
アプリケーション情報インスタンスを取得
S1003
資源回収インスタンスを1つ取り出す
S1004
アプリケーション情報
インスタンスを比較
一致
不一致
S1005
資源回収インスタンスに
確保した資源を記録する
S1006
全インスタンスを
比較終了
S1007
資源回収インスタンスを作成し
確保した資源とアプリケーション
情報インスタンスを記録する
S1008
資源回収インスタンスをアプリケーション
情報インスタンスに登録する
S1009
資源をアプリケーションに提供
END

【図１１】


START
S1101
アプリケーション情報インスタンスの
アプリケーション状態取得メソッドを呼び出し、
アプリケーションの状態取得
S1102
NO
資源を回収
YES
S1103
対応する資源を解放
S1104
資源回収インスタンスを解放
END

【図１２】


1201
空間
1202
1202
タスク
タスク
1203
1203
1203
1203
スレッド
スレッド
スレッド
スレッド
1206
1206
1206
1206
タスク資源
タスク資源
1205
1205
空間資源
1204

【書類名】　要約書

【要約】

【課題】　システムに新規のデバイスなどを追加する場合、資源を管理する方法をシステムに組み込むため、システム自体の修正・改善・追加をしなければならない。

【解決手段】　資源を供給するライブラリ１０４は、アプリケーションに資源を提供する際、資源回収のタイミングを通知してもらうためコールバック関数を資源回収通知部１０３ｂに登録する。資源回収を行うタイミングで資源回収通知手段１０３ｂは登録されているコールバック関数を呼び出し、ライブラリは、この呼び出しに応じて資源を回収する。

【選択図】　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５８２１〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社